In the Claims:

Please amend claims 8 and 13, cancel claims 10-12, and add new claims 17-30, all as shown below.

1 – 7. (Canceled)

8. (Currently Amended): A method for protection against interleaving transactions, comprising:

communicating with a resource manager from an application using an Application Programming Interface (API), wherein ~~the API~~ multiple threads of the application utilize ~~utilizes~~ a shared logical connection to the resource manager;

controlling transaction demarcation using the Java™ Transaction API (JTA);

communicating with the resource manager from a first thread of a transaction manager during two phase commit processing using an XAResource interface;

enlisting a resource, wherein the first thread of the transaction manager associates a unique transaction identifier with work that is performed on the [[a]] resource by invoking XAResource.start() on the resource and subsequent application updates to the resource are associated with a global transaction, wherein the resource is wrapped in an object that the transaction manager uses to synchronize concurrent enlistment requests, and wherein the transaction manager maintains a collection of wrapped XAResource objects which is consulted on each resource enlistment;

delisting a resource, wherein the first thread of the transaction manager invokes XAResource.end() on the resource and future application updates on the resource over the shared logical connection are disassociated from the global transaction; and

blocking a second thread of the transaction manager from calling XAResource.start() on the resource until the first thread of the transaction manager has called XAResource.end() on the resource.

9. (Previously Presented): The method of claim 8, wherein the application communicates to the resource manager using JDBC™ or JMS.

10 - 12. (Canceled)

13. (Currently Amended): The method of claim 8 ~~12~~, wherein each request to enlist the resource will first check to see if there is a lock being held on the resource by another thread of control.

14. (Previously Presented): The method of claim 13, wherein if a resource is not yet locked, a lock is granted to an accessor and held until the owner of the transaction ID delists the resource.

15. (Previously Presented): The method of claim 14, wherein waiting threads are signaled when a lock is freed and one of the waiting threads will be granted the lock and allowed to proceed with its enlistment.

16. (Previously Presented): The method of claim 15, wherein the collection of wrapped XAResource objects is periodically garbage collected to clear stale and unused entries.

17. (New): The method of claim 8, wherein the transaction manager maintains an enlistment data structure to manage the collection of wrapped XAResource objects.

18. (New): The method of claim 17, wherein the transaction manager contains identification information regarding the resource within the enlistment data structure at least until the transaction manager receives a signal indicating the resource has generated a result or completed a service invoked by the application.

19. (New): The method of claim 18, wherein at a predetermined time interval, each enlistment of a resource that has not been accessed for a certain period of time is delisted from the enlistment data structure.

20. (New): The method of claim 18, wherein a resource is delisted from the enlistment data structure when it has not been accessed for a certain period of time.

21. (New): A computer-readable storage medium, storing instructions for protection against interleaving transactions, the instructions comprising:

communicating with a resource manager from an application using an Application Programming Interface (API), wherein multiple threads of the application utilize a shared logical connection to the resource manager;

controlling transaction demarcation using the Java™ Transaction API (JTA);

communicating with the resource manager from a first thread of a transaction manager during two phase commit processing using an XAResource interface;

enlisting a resource, wherein the first thread of the transaction manager associates a unique transaction identifier with work that is performed on the resource by invoking XAResource.start() on the resource and subsequent application updates to the resource are associated with a global transaction, wherein the resource is wrapped in an object that the transaction manager uses to synchronize concurrent enlistment requests, and wherein the transaction manager maintains a collection of wrapped XAResource objects which is consulted on each resource enlistment;

delisting a resource, wherein the first thread of the transaction manager invokes XAResource.end() on the resource and future application updates on the resource over the shared logical connection are disassociated from the global transaction; and

blocking a second thread of the transaction manager from calling XAResource.start() on the resource until the first thread of the transaction manager has called XAResource.end() on the resource.

22. (New): The computer-readable storage medium of claim 21, wherein the application communicates to the resource manager using JDBC™ or JMS.

23. (New): The computer-readable storage medium of claim 21 wherein each request to enlist the resource will first check to see if there is a lock being held on the resource by another thread of control.

24. (New): The computer-readable storage medium of claim 23, wherein if a resource is not yet locked, a lock is granted to an accessor and held until the owner of the transaction ID delists the resource.

25. (New): The computer-readable storage medium of claim 24, wherein waiting threads are signaled when a lock is freed and one of the waiting threads will be granted the lock and allowed to proceed with its enlistment.

26. (New): The computer-readable storage medium of claim 25, wherein the collection of wrapped XAResource objects is periodically garbage collected to clear stale and unused entries.

27. (New): The computer-readable storage medium of claim 21, wherein the transaction manager maintains an enlistment data structure to manage the collection of wrapped XAResource objects.

28. (New): The computer-readable storage medium of claim 27, wherein the transaction manager contains identification information regarding the resource within the enlistment data structure at least until the transaction manager receives a signal indicating the resource has generated a result or completed a service invoked by the application.

29. (New): The computer-readable storage medium of claim 28, wherein at a predetermined time interval, each enlistment of a resource that has not been accessed for a certain period of time is delisted from the enlistment data structure.

30. (New): The computer-readable storage medium of claim 28, wherein a resource is delisted from the enlistment data structure when it has not been accessed for a certain period of time.